Writing R functions

A first, simple R function ***For course #3***

```
print.name = function( )
{
name = ' write your name using Roman characters '
cat('My name is: ', name, '\n')
# Equivalent statement: cat("My name is: ", name, "\n")
}

# Usage:  print.name( )
```

A function that does more: add two numbers

```
add.numbers = function(a, b)
{
c = a+b
cat('c = ', c, '\n')
return(c)
}

# Usage:  toto = add.numbers(32, 183)
          toto
```

A variant: add and subtract two numbers

```
add.numbers2 = function(a, b)
{
c = a+b
d = a-b
cat('c = ', c, '  d = ', d, '\n')
return(list(plus=c, minus=d))
}
```

Note: the name of an element in the return list does not have to be identical to the name of the corresponding variable in the function.

```
# Usage:  toto2 = add.numbers2(32, 183)
          toto2
          summary(toto2)
          toto2$plus
          toto2$minus
```

Write on a single line a function that computes the length of a vector.

Write on a single line a function that normalizes a vector (eq. 2.7).

An R function for Principal Component Analysis (PCA) ***For course #8***

Use the matrix algebra from pp. 5-6 of the "Practicals_in_R" to write your own R function for PCA. The data file will have been read to the R console when you will use your function. So, the 'read.table' command should not be included in your function.

Work in a text file. First, give a name of your choice to your function, as we did in previous exercises. Open the curly parenthesis {. Then, start copying from the following line on p. 5:

```
Y.mat = as.matrix(Y)
```

down to the following line found on p. 6:

```
G = F %*% diag(Y.eig$values^(-0.5))
```

Run each line of code in the R console in order to understand its role. Think about the effect of each line of code with respect to the PCA algebra studied in class. Eliminate all redundant and unnecessary lines.

For the graph, use the following function: `biplot(F,U)`

Your function must produce the following list of output elements:
`return(list(eigen.val=Y.eig$values, U=U, F=F, Usc2=U2, G=G))`

Close the curly parenthesis to indicate the end of the function.

When you have checked that the basic function works correctly, incorporate the following options:

(1) Use unstandardized or standardized data (**Y** must be centered in all cases)

```
# The first line of the fuction must be like the following:
name.of.your.pca.function = function(Y, scaling.type=1, standardize=FALSE)

# Modify the centring line as follows to incorporate the scaling option:
Y.cent = scale(Y.mat, center=TRUE, scale=standardize)
```

(2) Option for scaling type 1 or type 2

```
if(scaling.type == 1) {
  # What do you have to do if this condition is true?
  biplot(F,U)
  } else {
  # What do you have to do if this condition is false?
  biplot(G,U2)
  }
```